2. Activités et Modèles de développement en Génie Logiciel

Bernard ESPINASSE

Professeur à l'Université d'Aix-Marseille

Plan

- **Les Activités du GL**
  - Analyse des besoins
  - Spécification globale
  - Conceptions architecturale et détaillée
  - Programmation
  - Gestion de configurations et intégration
  - Validation et vérification
- **Les Modèles de développement du GL**
  - Le modèle de la cascade
  - Le modèle en V
  - Le modèle en spirale
  - Modèles par incréments

Les activités nécessaires au développement du logiciel

• maintenant définies de façon plus **précise**

• utilisent et produisent des **documents** (textes, programmes, traces d'exécution, etc.)

• ont **plus ou moins d'importance** selon le processus de développement retenu et la nature du logiciel à produire (peuvent être inutiles dans certains cas)

Ces grandes activités sont :

- **Analyse des besoins**
- **Spécification globale**
- **Conception architecturale et détaillée**
- **Programmation**
- **Gestion de configurations et intégration**
- **Validation et vérification**

Quelques chiffres…d'efforts et de coûts

Répartition générales des efforts :

% de l'effort total de développement d'un logiciel :

- **programmation** : **10 à 20%**
- **spécification** et **conception** : environ **40%**
- **validation** et **vérification** : de l'ordre de **40%**

Coûts selon les domaines d'application :

Domaines d'application	Analyse / Conception	Réalisation	Test
de commande	46	20	34
embarqués	34	20	46
d'exploitation	33	17	50
scientifique	44	26	30
de gestion	44	28	28

Analyse des besoins

Objet

- **éviter** de développer un logiciel non adéquat
- menée en liaison avec les **études de faisabilité** et la **planification**
- étude du **domaine d'application** : états **actuel** et **futur** de l'environnement du système, déterminer les **frontières**, le **rôle**, les **ressources disponibles** et **requises**, les **contraintes d'utilisation** et de **performance**, etc.

Données

- fournies par des experts/utilisateurs du domaine d'application

  -> établir un **dialogue** entre informaticiens et experts/utilisateurs du domaine

- méthodes utilisées : relèvent plutôt des **sciences cognitives** : entretiens, questionnaires, observations de l'existant, études de situations similaires

Résultat

- ensemble de documents décrivant l'environnement du futur système, son rôle et sa future utilisation (parfois manuel d'utilisation préliminaire)

Spécification globale

Objet

- établir une première description du futur système, corrélée avec :
  - l'analyse des besoins (souvent regroupées dans même étape)
  - la validation

Données

- résultats de l'analyse des besoins + considérations technique/faisabilité informatique

Résultat : spécification technique de besoins STB

- **le quoi, pas le comment :**
  - une description de ce que doit faire le logiciel en évitant des décisions prématurées de réalisation
  - trop difficile d'anticiper leurs conséquences sur la réalisation finale en termes de performances, ressources, ou même de faisabilité.
- point de départ au développement
- souvent : 1° version du manuel de référence + compléments au manuel d'utilisation.

Conceptions architecturale et détaillée

Objet

• une **description du logiciel** très proche d'un programme avec détails d'implémentation,

2 étapes :

1 - Conception architecturale :

- décomposer le logiciel en composants plus simples
- préciser les interfaces et les fonctions de chaque composant
- fournir une description de l'architecture du logiciel et un ensemble de spécifications de ses divers composants.

2 - Conception détaillée :

- fournir pour chaque composant une description précisant comment ses fonctions sont réalisées : algorithmes, représentation des données,...

Remarques

• **frontière entre spécification et conception souvent floue** car pas raisonnable de spécifier un système indépendamment de toute considération de faisabilité.

• la **conception** commence souvent pendant la **spécification**, et peut la remettre en cause

• des **contraintes de réalisation** peuvent anticiper sur la conception au moment de la spécification

Programmation

- passer du résultat de la conception détaillée à **un ensemble de programmes ou de composants de programmes**
- la mieux **maîtrisée** et la mieux "**outillée**" (**parfois automatisée**)

Gestion de configurations

- permettre la **gestion des composants** du logiciel, d'en maîtriser **l'évolution** et les **mises à jour** tout au long du processus de développement (documentation homogène)

Intégration

- **assembler** tout ou partie des **composants logiciels** pour obtenir un système exécutable
- existe souvent **plusieurs choix possibles** pour certains composants -> variantes du logiciel (ex: pour des systèmes d'exploitation différents)
- utilise la **gestion de configuration** pour :
    - assembler des **versions cohérentes de composants**
    - gérer des **variantes du logiciel**

Validation

***a-t-on décrit le "bon" système**, c'est-à-dire un système qui répond à l'attente des utilisateurs et aux contraintes de leur environnement ?*

- s'assurer de l'adéquation des résultats de l'analyse des besoins et de la spécification globale
- consiste en des **revues et inspections de spécifications** ou de **manuels**, et du **prototypage rapide**

Vérification

***le développement est-il correct** par rapport à la spécification de départ ?*

- **s'assurer** que les **descriptions successives du logiciel**, et, in fine, le logiciel lui-même, **satisfont la spécification globale** : **inspections** de **spécifications**, de **programme**, **preuve** et **tests**.
- **preuve** : porte sur une spécification détaillée ou un programme et permet de prouver que celle-ci ou celui-ci satisfait bien la spécification de départ.
- **test** : consiste à rechercher des erreurs dans une spécification ou un programme par :
  - examen ou analyse du texte (**test statique**)
  - par des exécutions sur sous-ensemble fini de données (**test dynamique**) :
    - **test unitaire** : tester des composants isolés ;
    - **test d'intégration** : tester un ensemble de composants vennant d'être assemblés
    - **test système** : tester le système sur son futur site d'exploitation, dans des conditions opérationnelles et au-delà (surcharge, défaillances matérielles, ...).

Rôle du maquettage (ou prototypage rapide)

- principale difficulté en validation = l'imprécision des besoins et des caractéristiques du système à développer

- développer très rapidement un programme, la **maquette**, qui est une ébauche du futur système et de préciser les besoins

  - elle n'en a **pas les performances**,

  - **ni toutes les fonctionnalités** et

  - elle **ne répond pas aux exigences de qualité** d'un produit fini.

- **maquette exploratoire** : soumise à des scénarios en liaison avec les futurs utilisateurs afin de préciser leurs besoins ou leurs souhaits

- **maquette expérimentale** : lors d'une étape de conception, permet l'expérimentation et la comparaison de choix différents

Important de **bien définir les objectifs d'une opération de prototypage rapide**, et d'en tenir compte pour la conception de la maquette

Le Modèle de la Cascade (waterfall model)

• modèle **très simple**, nécessite d'avoir un certain nombre d'étapes (Boehm 76)

première version:

Faisabilité

Analyse des besoins et planification

Conception de produit

Conception détaillée

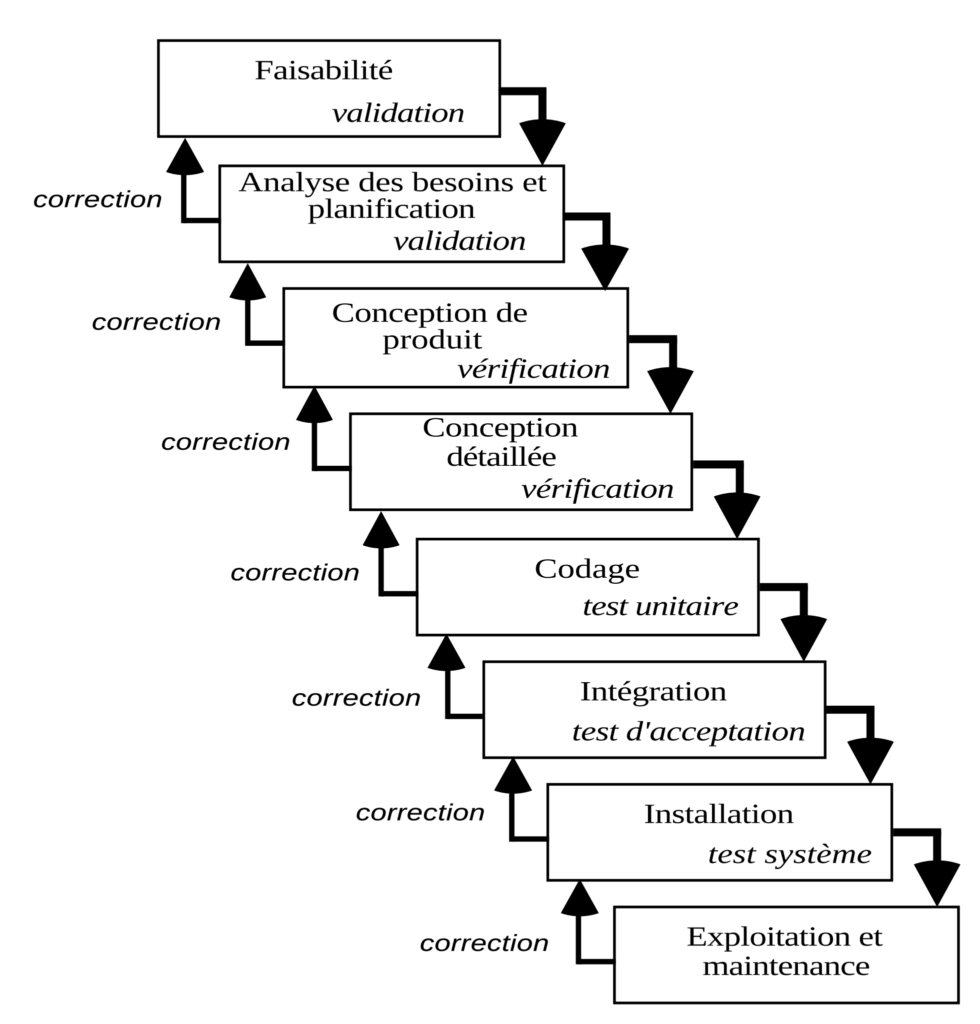
Codage

Intégration

Installation

Exploitation et maintenance

versions actuelles :



Le Modèle de la Cascade (waterfall model)

- une étape doit se terminer à une **certaine date**, par la production de certains **documents** ou **logiciels**.
- les **résultats de l'étape** sont soumis à une **revue approfondie**, et on ne passe à l'étape suivante que **quand ils sont jugés satisfaisants**.
- certaines **étapes** portent le nom d'une **activité** essentielle pour cette étape, mais n'impose pas qu'elle n'ait lieu que dans cette étape.
- d'**autres activités interviennent** : le **contrôle technique** ou la **gestion de configurations** présents **tout au long du processus**.
- les **flèches ascendantes** (versions actuelles) expriment qu'une étape ne remet en cause que l'étape précédente:

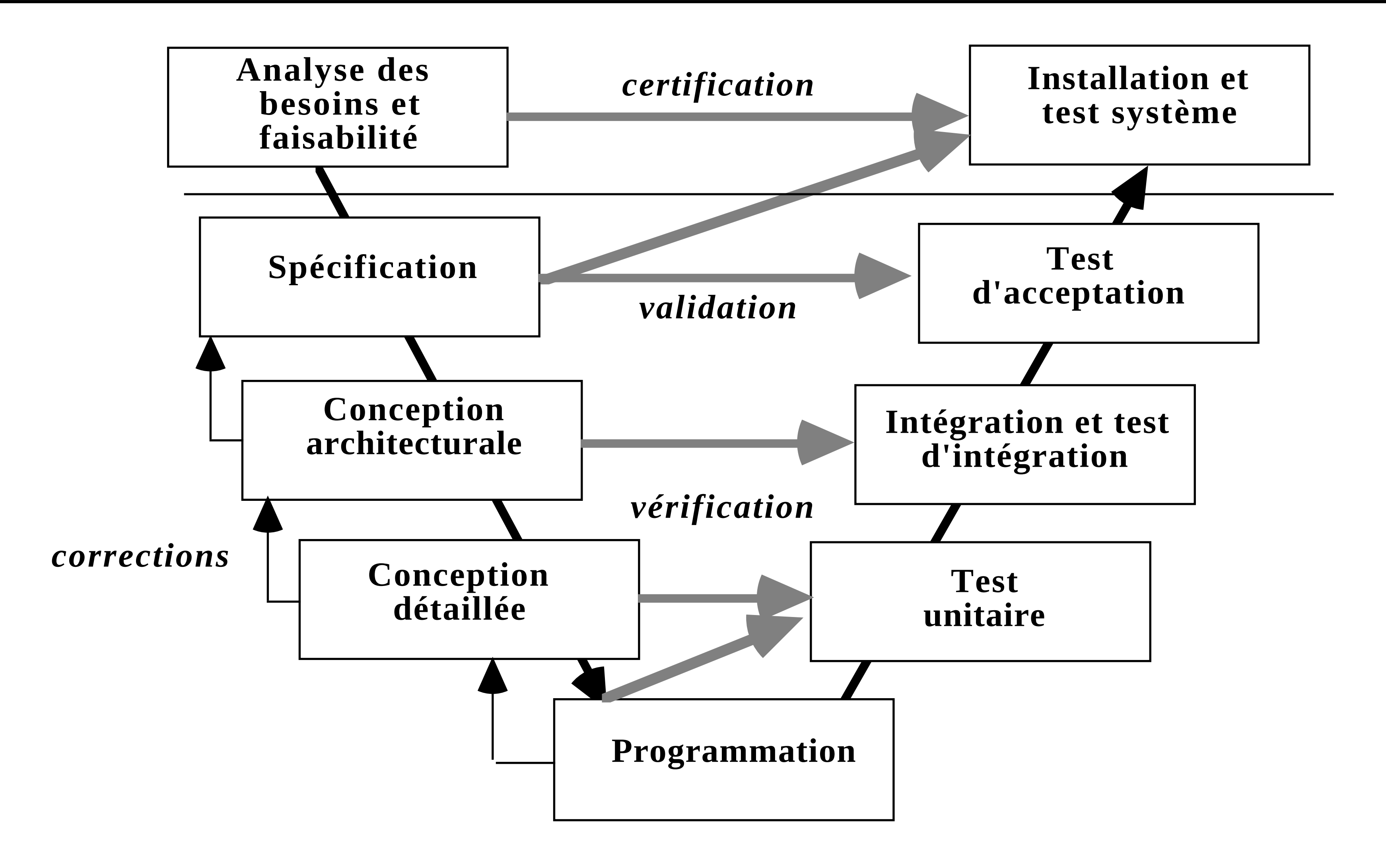
  -> **en pratique**: souvent un **voeu pieux**, il y a toujours des problèmes qui se **propagent de bas en haut**

- **documents, normes, recommandations** décrivent précisément les étapes (IEEE, AFNOR).

Limites du modèle

souvent abandonné au profit du **modèle en V**, plus réaliste dans l'articulation entre les activités de **réalisation** et de **validation/vérification**.

le Modèle en V



• les premières étapes du développement (**conception logiciel**) prépare les dernières étapes (**validation et vérification**)

le Modèle en V

- **2 sortes de dépendances entre étapes** :
  - **celles du V** : enchaînement et l'itération éventuelle du modèle de la cascade les étapes se déroulent **séquentiellement** en suivant **le V de gauche à droite**
  - **celles transversales** : une partie des **résultats** de **l'étape de départ** est utilisée directement par l'**étape d'arrivée**

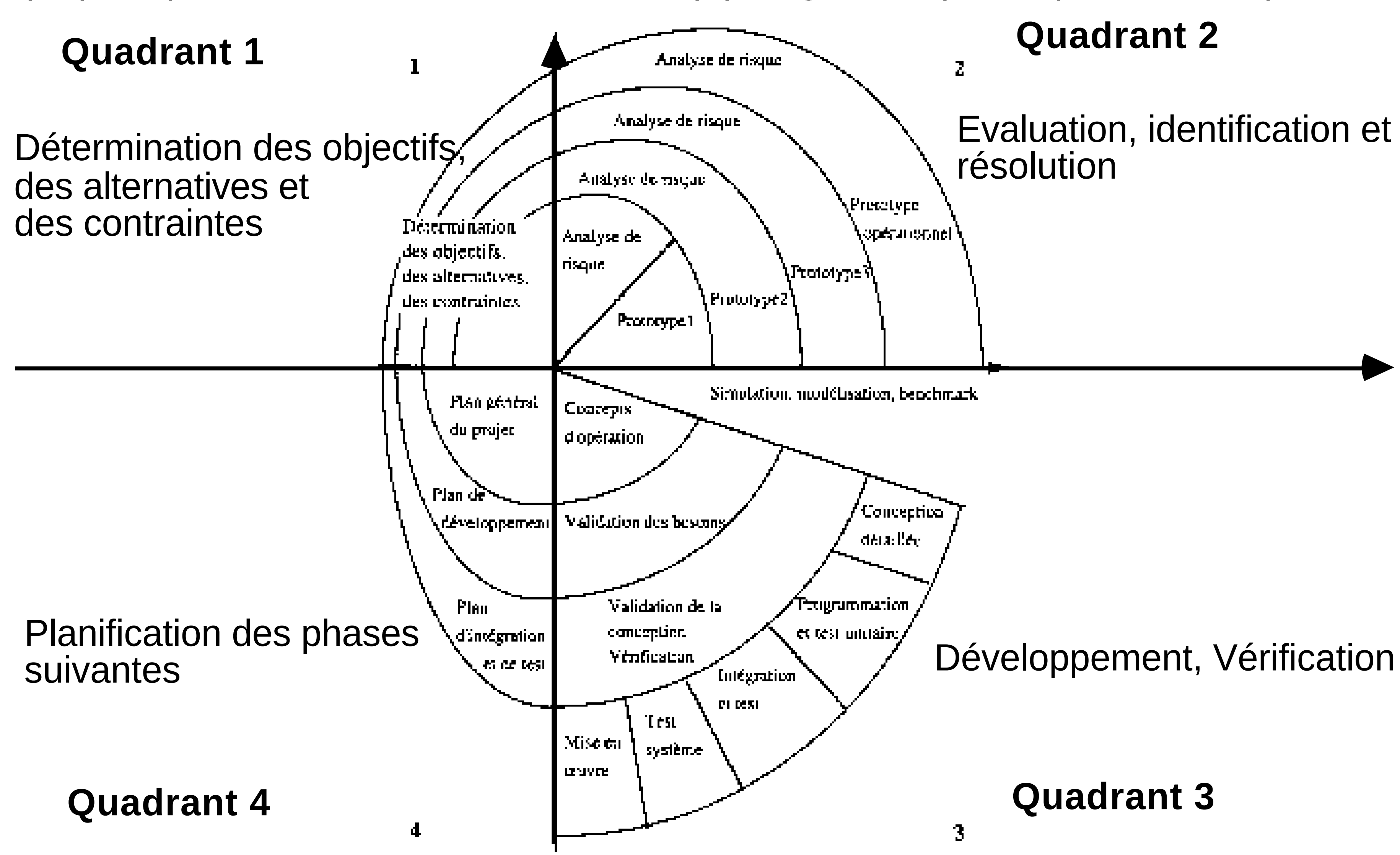
exemple:

à l'issue de la ***conception architecturale****, le protocole* ***d'intégration*** *et les jeux de* ***test d'intégration*** *doivent être complètement décrits.*

- évite d'énoncer une **propriété impossible à vérifier** objectivement une fois le logiciel réalisé:
  - avec toute **décomposition** doit être décrite la **recomposition**,
  - toute **description d'un composant** est accompagnée des **tests** qui permettront de s'**assurer qu'il correspond à sa description**

le Modèle de la Spirale

• proposé par B.BOEHM en 1968, beaucoup plus général que les précédent et peut les inclure



le Modèle de la Spirale: les phases en détail

- met l'accent sur l'**analyse de risque**
- chaque **cycle de la spirale** se déroule en 4 phases représentées par **4 quadrants** :
  - 1. Détermination :
    - des **objectifs du cycle**,
    - des **alternatives** pour les atteindre,
    - des **contraintes**, à partir des **résultats des cycles précédents**, ou, si il n'y en a pas, d'une **analyse préliminaire des besoins**
  - 2. **Analyse des risques**, **évaluation** des alternatives, éventuellement **maquettage**
  - 3. **Développement** et **vérification** de la solution retenue
  - 4. **Revue des résultats** et **planification du cycle suivant.**
- **quadrant 3** correspond à un développement classique (ou portion) et un des modèles précédents (de la cascade ou en V) peut s'appliquer
- ce **"super" modèle** permet d'encadrer le développement proprement dit par des phases consacrées à la **détermination des objectifs** et à l'**analyse de risque**.

Modèle en Spirale: risques majeurs du développement de logiciel

- fournit **liste de risques** encourus dans développement de logiciel et suggére des **solutions**:
  - **Défaillance de personnel** : embauche de personnel de haut niveau; adéquation entre profil et fonction ; esprit d'équipe ; formation mutuelle ; personnes clés.
  - **Calendrier et budget irréalistes** : estimation détaillée des coûts et calendriers développement incrémentai ; réutilisation ; élagage des besoins.
  - **Développement de fonctions inappropriées** : analyse de l'organisation, de la mission; revues d'utilisateurs; manuel d'utilisation précoce
  - **Développement d'interfaces utilisateurs inappropriées** : maquettage ; scénarios et revues d'utilisateurs ; analyse des tâches.
  - **Produit "plaqué or"** : élagage des besoins ; maquettage ; analyse des coûts bénéfices ; conception prenant en compte les coûts.
  - **Volatilité des besoins**: seuil élevé de modification ; masquage d'information développement incrémentai où les derniers incréments sont les plus changeants.
  - **Composants externes manquants** : inspections; essais/mesures; analyse de compatibilité
  - **Tâches externes défaillantes** : audit avant attribution de sous-traitance ; contrats avec bonus ; revues.
  - **Problèmes de performances**: simulations ; modélisations ; essais et mesures maquettage.
  - **Exigences démesurées par rapport à la technologie**: analyses techniques de faisabilité ; maquettage.

le Modèle de la Spirale: mise en oeuvre

- le **premier cycle** :
  - une **analyse préliminaire de besoins** affinée au cours des premiers cycles, en prenant en compte les **contraintes** et l'**analyse des risques**.
  - une utilisation systématique de **maquettes exploratoires**.
- les **cycles suivants** :
  - les **3°quadrants =** conception, les choix guidés par **maquettes expérimentales**.
- le **dernier cycle** se termine par la **fin d'un processus de développement classique**

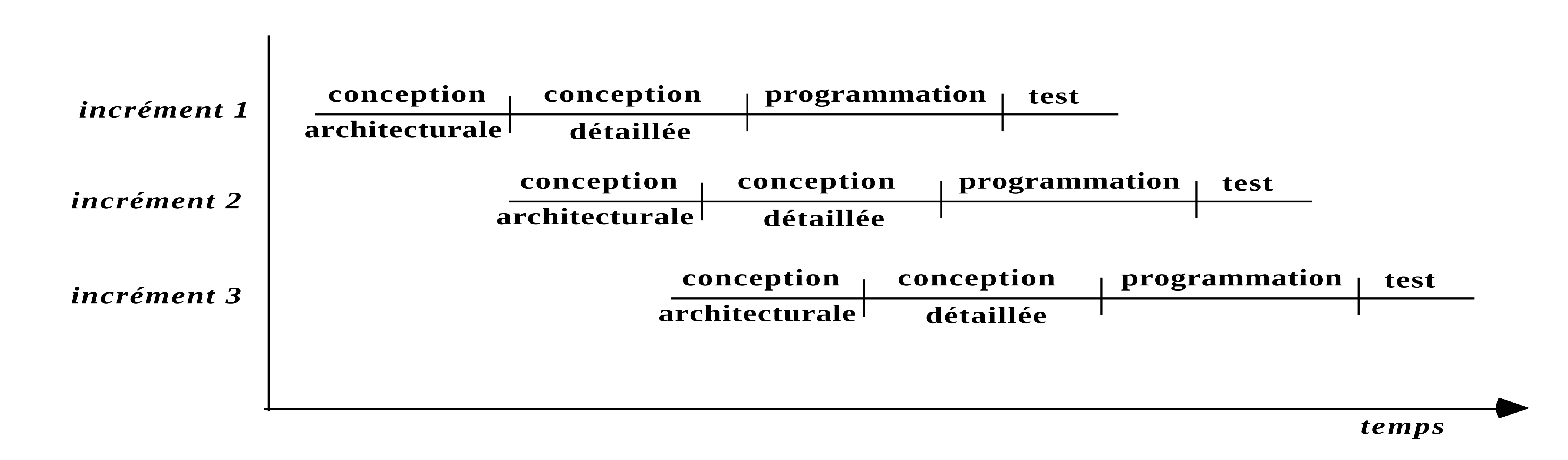
en conclusion

- mise en oeuvre demande des **compétences** et un **effort importants**
- **moins expérimenté** et **moins documenté** que les précédents .
- utilisation complète **adaptée à des projets innovants**, à risques, et dont les enjeux sont importants.

les Modèles par Incréments

- **modèles précédents** :
  - **décomposition en composants** (conception architecturale) puis
  - **composants développés indépendamment les uns des autres**, en parallèle ou en séquence, selon les ressources disponibles
- **modèles par incréments** :

  un seul sous-ensemble des composants est développé à la fois :
  - un **logiciel noyau** est tout **d'abord développé** puis
  - des **incréments** sont successivement **développés** et **intégrés**



Modèles par Incréments : avantages

- chaque **développement** est **moins complexe**
- les **intégrations** sont **progressives**
- **livraisons** et **mises en service** possibles **après chaque intégration** d'incrément.
- permet de **mieux lisser dans le temps l'effort** de développement **et les effectifs**.
- souvent utilisée pour de grands projets, fonctionnant par appels d'offres et sous-traitances.

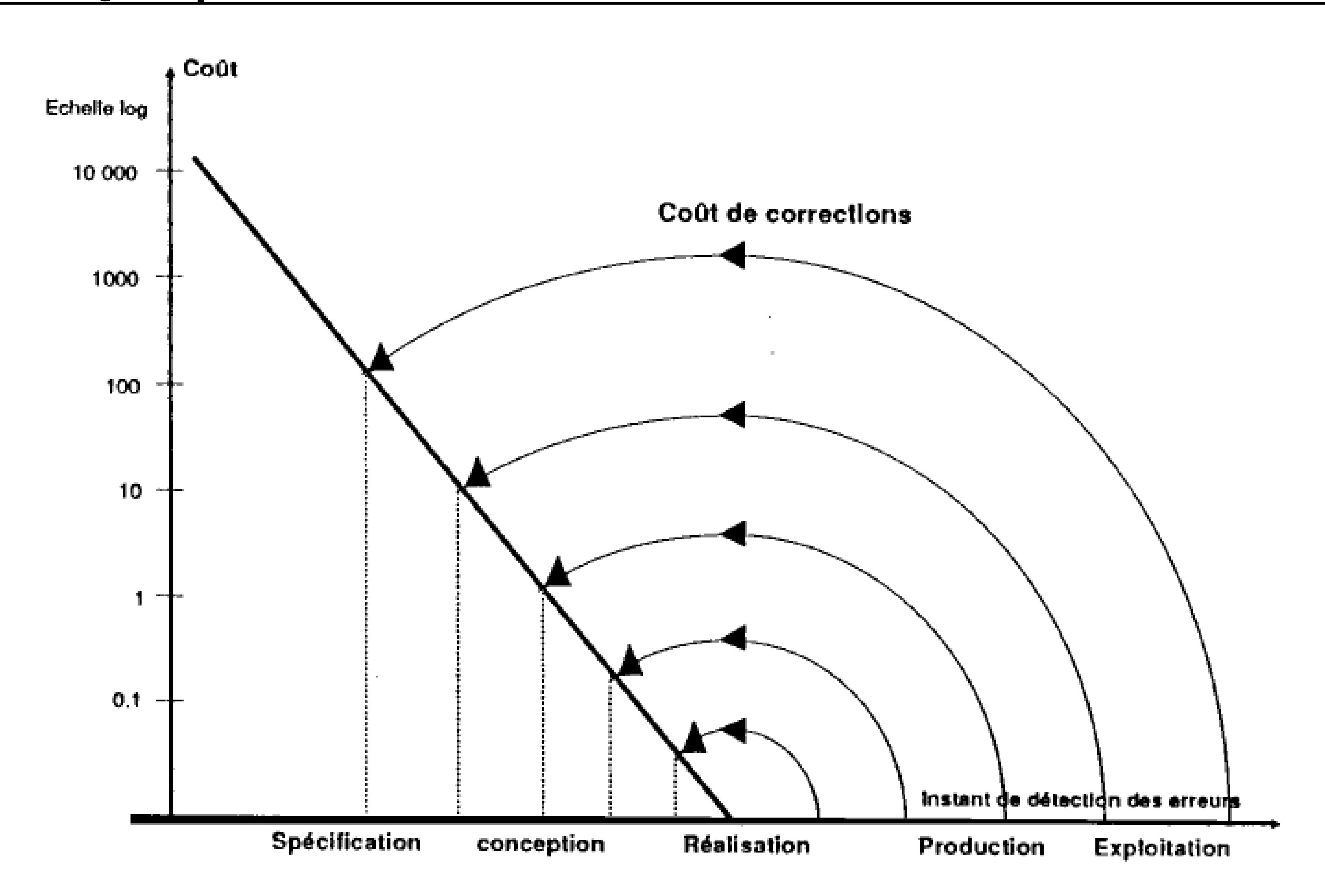
Modèles par Incréments : risques

- **risque majeur** : remise en cause du noyau ou les incréments précédents
- **autre risque** : être incapable d'intégrer un incrément.

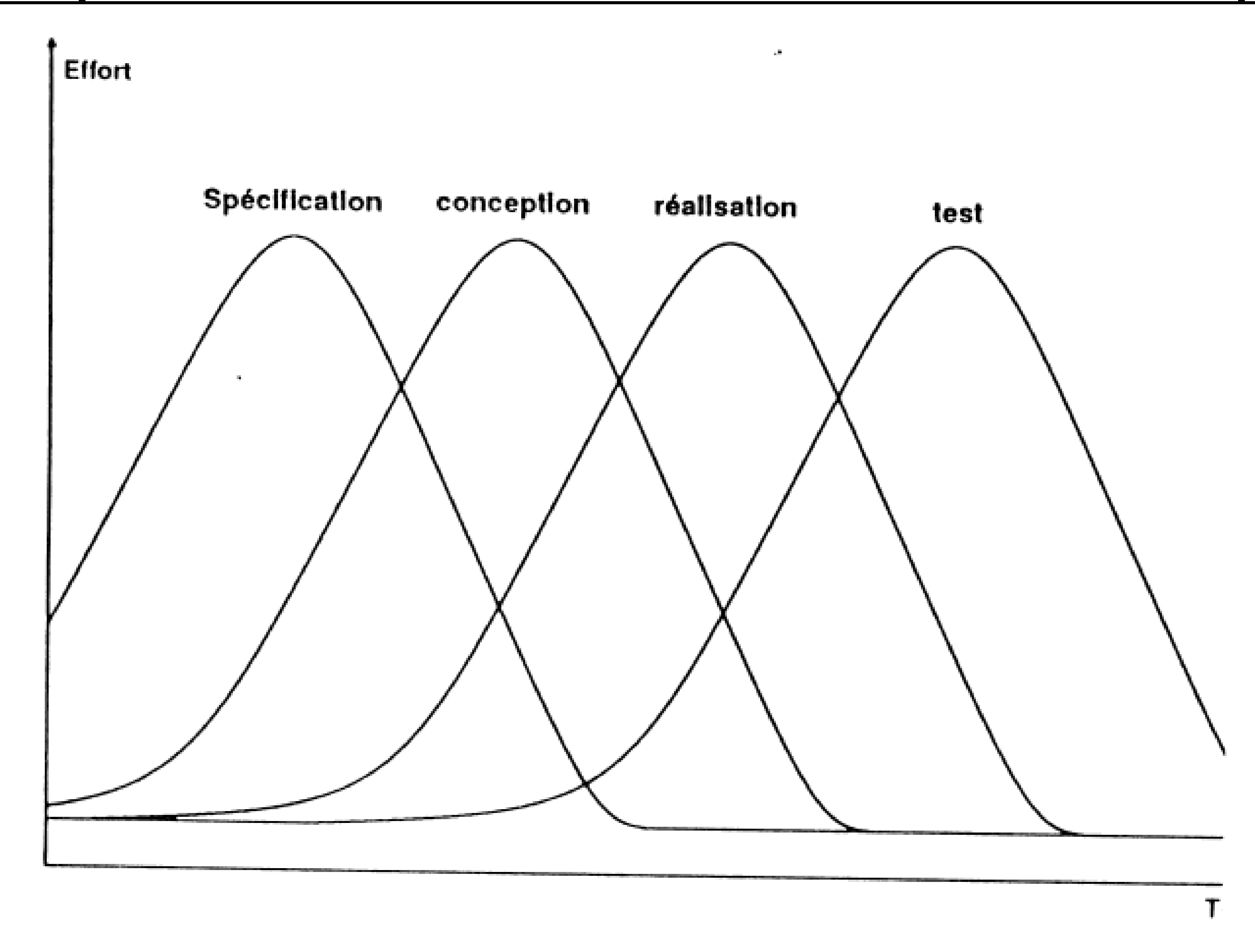
en conséquence :

- au **début du projet, spécification globale** du **noyau**, des **incréments**, et de leurs **interactions**
- **incréments aussi indépendants que possible** (aussi bien fonctionnellement qu'au niveau des calendriers de développement)

Quelques constatations: coût de correction des erreurs



Quelques constatations: recouvrement souhaitable des phases



Quelques constatations: facteurs affectant le coût du logiciel

